[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/883,508	06/19/2001	Jeffrey A. Bedell	53470.003042	8696

21967 7590 07/26/2006

HUNTON & WILLIAMS LLP
INTELLECTUAL PROPERTY DEPARTMENT
1900 K STREET, N.W.
SUITE 1200
WASHINGTON, DC 20006-1109

EXAMINER	
ZHEN, LI B	
ART UNIT	PAPER NUMBER
2194	

DATE MAILED: 07/26/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/883,508	BEDELL ET AL.	
	Examiner	Art Unit	
	Li B. Zhen	2194	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 May 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

[illegible]

WILLIAM THOMSON
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1 – 18 are pending in the application.

Response to Arguments

2. Applicant's arguments filed 05/03/2006 have been fully considered but they are not persuasive. In response to the Non-Final Office Action dated 02/03/2006, applicant argues:

(1) Mariani does not teach or suggest "determining using a computer processor an appropriate manner of executing the selected function". The word "manner" refers to the way that something is done, not the fact of whether or not it is done [pp. 8 – 9];

(2) Office Action has not show a single, coherent embodiment of Mariani that teaches all the elements of the claims. For instance, the "selected function" applies "modifying code" and "recompiling" [p. 9];

(3) Office action appears to apply "source code file" and "object code file" to a "selected object" [p. 9];

(4) Fontana does not teach the claimed limitations because updating dependent components in response to a user prompt is not the equivalent of "automatically identifying dependent objects, automatically causing appropriate functions to be performed on the dependent object and automatically causing the execution of the selected function on the selected object" [p. 10];

(5) Office has failed to provide any proper motivation for modifying Mariani [p. 11];

(6) Applicant submits that the meta model disclosed by Almond does not teach "the selected object is contained in metadata of an on-line analytical processing system" [p.13]; and

(7) Mariani system cannot be combined with Almond to copy objects in the way that it selectively recompile object code; thus, the combination of Mariani and Almond does not teach or suggest claim 6 [pp. 13-14].

As to argument (1), examiner respectfully disagrees and notes that Mariani teaches determining using a computer processor an appropriate manner of executing the selected function. Mariani's rebuilding of the executing program through recompiling of the object files corresponds to the claimed selected function. The minimal rebuild system doesn't just determine whether to execute the function or not. The recompiling function determining how the object code files are dependent on the header files and detecting changes to the header files [col. 9, lines 1 – 15 of Mariani]. Based on these results, the recompiling function determines which files to recompile [the object code files that have changed and the dependent files]. By determining the changed files and its dependent files, the recompile function determines a particular manner of rebuilding of the executable program.

As to argument (2), examiner notes that the modifying code is function performed by the class wizard that triggers a call to the minimal rebuild engine to perform a recompiling function [col. 10, lines 7 – 46]. Whenever the code is modified, the recompiling function is called. Therefore, it can be interpreted that the recompiling function is a part of the modification function. The modification command is not complete until the recompiling function is performed and the recompiling function is a sub process of the modification command.

In response to argument (3), examiner disagrees and notes that the source code file correspond to the selected object. The object code files are used to determine dependency by determining how object files are dependent on the header files. The recompiling function is performed on the source code files that correspond to the identified object code files. The result of compiling a source code file is an object code file [i.e. col. 8, lines 43 – 52].

As to argument (4), examiner disagrees and submits that Fontana's updating function corresponds to the claimed function. The claims only recite a function and the scope of the claims does not preclude the claimed function to read on Fontana's update function. For example, Fontana teaches a wrapper tool to develop interfaces for a component or any third party developed software that helps in the component build process [col. 4, lines 10 – 30], automatically identifying dependent objects [dependencies are then analyzed to identify dependent components (block 65); col. 7, lines 22 – 34], automatically causing appropriate functions to be performed on the dependent objects [if the user does want to update dependent components (yes leg of the diamond 66), then the dependent components are retrieved from the source control program (block 69); col. 7, lines 35 – 43] and automatically causing execution of the selected function on the selected object [dependent components are accordingly updated (block 76); col. 7, lines 43 – 58]. Therefore, the combination of Mariani and Fontana teaches applicant's invention as claimed.

In response to argument (5), the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, motivation can be found in col. 1, lines 33 – 37 of Fontana.

As to argument (6), applicant appears makes a submission but does not provide a detail argument of how the mapped elements of the prior art fail to teach the claimed limitation. Examiner disagrees and submits that the prior art teaches the claimed limitation. For example, Almond teaches an object contained in metadata [maps an object into a meta model which facilitates version control; col. 3, lines 1 – 54 and col. 6, lines 40 – 67] of an on-line analytical processing program [View reports--quickly view project activity status; col. 39, lines 15 – 39].

In response to argument (7), examiner disagrees and submits that copying objects and recompiling object code are separate functions and the functions do not affect each other; therefore the system are combinable and the motivation for combining the references can be found in col. 3, lines 3 – 10 of Almond.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 3 – 5, 10 – 12 and 16 – 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent NO. 5,854,932 to Mariani in view of U.S. Patent No. 6,167,563 to Fontana et al. [hereinafter referred to as Fontana, both references cited in the previous office action].

5. As to claim 1, Mariani teaches the invention substantially as claimed including a computer implemented method for managing groups of objects [source code files 60 and header files] for use in a reporting system project [a development environment 52 which includes various tools 54 for creating, editing, and compiling source code files 60 and header files 62 for a user's programming project; col. 7, line 60 – col. 8, line 7] comprising the steps of:

receiving a command [modify] to perform a selected function on a selected object [With the editors 70, the user can directly modify or add C++ statements to the source code files 60 and header files 62; col. 8, lines 7 – 22];

identifying using a computer processor dependent objects referred to by the selected object [for each of the object code files 82...the minimal rebuild system 100 generates dependency information...consisting of a set of classes...on which the respective object code file depends, and a set of dependency types...of the object code file on each class in the set; col. 12, lines 27 – 67];

determining using a computer processor an appropriate manner of executing the selected function [minimal rebuild system 100 determines when recompiling can be avoided by determining how the object code files 82 are dependent on the header files 62; col. 9, lines 1 – 15];

determining using a computer processor appropriate functions to be performed on the dependent objects [minimal rebuild system 100 selectively recompiles the source code files 60...so as to detect changes to all header files 62...that were changed since the last build of the project. The minimal rebuild system 100 then utilizes the detected changes to the header files 62 to determine which of the remaining source code files 60 to recompile and which recompiles can be avoided; col. 17, lines 18 – 30];

causing the appropriate functions to be performed on the dependent objects [omit compiling selected source file and resave object file, step 174, Fig. 6B; compile selected source file into object file, step 176, Fig. 6B; col. 19, lines 33 – 67]; and

causing the execution of the selected function on the selected object in the appropriate manner [minimal rebuild system 100 first recompiles the source code files that were changed since the user's project was last built; col. 17, lines 30 – 43].

6. Although Mariani teaches the invention substantially as claimed, Mariani does not specifically teach automatically identifying dependent objects, automatically causing appropriate functions to be performed on the dependent objects and automatically causing execution of the selected function on the selected object.

However, Fontana teaches a wrapper tool to develop interfaces for a component or any third party developed software that helps in the component build process [col. 4, lines 10 – 30], automatically identifying dependent objects [dependencies are then analyzed to identify dependent components (block 65); col. 7, lines 22 – 34], automatically causing appropriate functions to be performed on the dependent objects [if the user does want to update dependent components (yes leg of the diamond 66), then the dependent components are retrieved from the source control program (block 69); col. 7, lines 35 – 43] and automatically causing execution of the selected function on the selected object [dependent components are accordingly updated (block 76); col. 7, lines 43 – 58].

7. It would have been obvious to a person of ordinary skill in the art at the time of the invention to apply the teaching of automatically identifying dependent objects, automatically causing appropriate functions to be performed on the dependent objects and automatically causing execution of the selected function on the selected object as taught by Fontana to the invention of Mariani because this provides a method for building or modifying software components inside a computer system and updating all dependent components automatically in a manner transparent to the user and the computer system [col. 1, lines 33 – 37 of Fontana].

8. As to claim 10, this is a system claim that corresponds to method claim 1; note the rejection to claim 1 above, which also meet this system claim.

9. As to claim 18, this is a product claim that corresponds to method claim 1; note the rejection to claim 1 above, which also meet this product claim.

10. As to claim 3, Mariani teaches the step of causing the appropriate functions to be performed on the dependent objects is performed prior to the step of causing the execution of the selected function in the appropriate manner [minimal rebuild system 100 reorders this list to place any source code files 60 that have changed since the user's project was last rebuilt first in the list; col. 17, lines 43 – 65].

11. As to claim 4, Mariani teaches the objects are grouped in projects and the selected function relates to manipulating objects within and between projects [For use in creating and editing the source code files 60 and header files 62 of the user's project, the development tools 54 of the environment 52 comprise one or more editors 70, and automated source code generators 72; col. 8, lines 7 – 22] and wherein within each project each object has a unique identifier [search for a name in the scope of the class; col. 13, lines 25 – 50] and a version identifier [date and time of the last change made to the header files; col. 10, lines 8 – 45].

12. As to claim 5, Mariani teaches comparing dependent objects at a source and objects at a destination to determine whether an object at the destination exists in an identical form to each of the dependent objects at the source and whether an object at the destination exists in a modified form to each of the dependent objects at the source [minimal rebuild system 100 preferably determines which of the header files 62 were changed since the project was last built by comparing the date stamps of the header

files to their last known date stamps; col. 18, lines 34 – 48] and determining, based on the step of comparing, which dependent object to copy from the source to the destination such that the selected object remains complete after execution of the selected function in the appropriate manner [the minimal rebuild system 100 selects the next source code file in the list that is not yet recompiled. At step 171, the minimal rebuild system 100 checks whether the object code file that was compiled in a previous build of the project from the selected source code file still exists; col. 19, lines 7 – 32].

13. As to claim 11, Mariani teaches the operational module interfaces [classes for interfacing with database management systems; col. 8, lines 23 – 43] with projects that reside in various environments [Class dependency information is similarly reduced for manually generated projects that use a class library; col. 12, lines 28 – 67].

14. As to claim 12, this is rejected for the same reasons as claim 3 above.

15. As to claims 16 and 17, these are rejected for the same reasons as claim 5 above.

16. Claims 2, 6 – 9 and 13 – 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mariani and Fontana further in view of U.S. Patent NO. 6,112,024 to Almond [cited in the previous office action].

17. As to claim 2, Mariani as modified does not teach the selected object is contained in metadata of an on-line analytical processing system.

However, Almond teaches an object cycle versioning system [col. 2, lines 39 – 52] and an object contained in metadata [maps an object into a meta model which facilitates version control; col. 3, lines 1 – 54 and col. 6, lines 40 – 67] of an on-line analytical processing program [View reports--quickly view project activity status; col. 39, lines 15 – 39].

18. It would have been obvious to a person of ordinarily skilled in the art at the time of the invention to apply the teaching of storing an object in metadata of an on-line analytical processing system as taught by Almond to the invention of Mariani as modified because this map objects to representations and allow operations supported by the system for versioning will execute correctly even if the objects are stored in a format other than a relational database, such as an object-oriented database, a file server, or other storage system [col. 3, lines 3 – 10 of Almond].

19. As to claim 6, Mariani as modified teaches the step of receiving is a step of receiving a command to copy a selected object from a source project to a destination project [Get--copy one or multiple objects to the user's local directory; col. 39, lines 16 – 39 of Almond].

20. As to claim 7, Mariani as modified teaches the unique identifier [search for a name in the scope of the class; col. 13, lines 25 – 50 of Mariani] and version identifier

[date and time of the last change made to the header files; col. 10, lines 8 – 45 of Mariani] of objects in the source project are similar to the unique identifier and version identifier of objects in the destination project [minimal rebuild system 100 preferably determines which of the header files 62 were changed since the project was last built by comparing the date stamps of the header files to their last known date stamps; col. 18, lines 34 – 48 of Mariani].

21. As to claim 8, Mariani as modified teaches comparing unique identifiers [search for a name in the scope of the class; col. 13, lines 25 – 50 of Mariani] and version identifiers [date and time of the last change made to the header files; col. 10, lines 8 – 45 of Mariani], whether the selected object exists in the destination project in an identical form and whether the selected object exists in the destination project in a modified form [minimal rebuild system 100 preferably determines which of the header files 62 were changed since the project was last built by comparing the date stamps of the header files to their last known date stamps; col. 18, lines 34 – 48 of Mariani]; and

selecting whether to copy the selected object from the source project to the destination project [Get--copy one or multiple objects to the user's local directory; col. 39, lines 16 – 39 of Almond], to replace an object in the destination project with the selected object [compile selected source file into object file, step 176, Fig. 6B; col. 19, lines 33 – 67 of Mariani], and to keep an object in the destination project as is [omit compiling selected source file and resave object file, step 174, Fig. 6B; col. 19, lines 33 – 67 of Mariani].

22. As to claim 9, Mariani as modified teaches the step of selecting includes prompting the user for a selection [user can directly modify or add C++ statements to the source code files 60 and header files 62; col. 8, lines 7 – 23 of Mariani].

23. As to claim 13, Mariani as modified teaches the operation module interfaces [RPC interface allows the system to surface an Object Cycle API…for development system clients; col. 2, lines 39 – 54 of Almond] with projects of an on-line analytical processing system [View reports--quickly view project activity status; col. 39, lines 15 – 39 of Almond].

24. As to claim 14, Mariani as modified teaches the operational module, upon receiving a user command to copy the selected object from a source project to a destination project [Get--copy one or multiple objects to the user's local directory; col. 39, lines 16 – 39 of Almond], determines, by comparing the unique identifiers [search for a name in the scope of the class; col. 13, lines 25 – 50 of Mariani] and the version identifiers [date and time of the last change made to the header files; col. 10, lines 8 – 45 of Mariani], whether the selected object exists in the destination project in an identical form and whether the selected object exists in the destination project in a modified form [minimal rebuild system 100 preferably determines which of the header files 62 were changed since the project was last built by comparing the date stamps of the header files to their last known date stamps; col. 18, lines 34 – 48 of Mariani].

25. As to claim 15, Mariani as modified teaches the operational module communicates with the user interface to select whether to copy the selected object from the source project to the destination project [Get--copy one or multiple objects to the user's local directory; col. 39, lines 16 – 39 of Almond], to replace an object in the destination object with the selected object [compile selected source file into object file, step 176, Fig. 6B; col. 19, lines 33 – 67 of Mariani], and to keep an object in the destination project as is [omit compiling selected source file and resave object file, step 174, Fig. 6B; col. 19, lines 33 – 67 of Mariani].

Conclusion

26. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

CONTACT INFORMATION

27. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Li B. Zhen whose telephone number is (571) 272-3768. The examiner can normally be reached on Mon - Fri, 8:30am - 5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, William Thomson can be reached on 571-272-3718. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Li B. Zhen
Examiner
Art Unit 2194

LBZ

[illegible]
WILLIAM THOMSON
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100